Міністерство освіти і науки України

Національний технічний університет України

“Київський політехнічний інститут”

Кафедра АСОІУ

**ЗВІТ**

про виконання комп’ютерного практикуму № 1

з дисципліни

“ Програмні додатки з використанням баз даних”

по темі

«Аналітичні запити»

Варіант 6

|  |  |  |
| --- | --- | --- |
| Прийняв: |  | Виконав: |
| Cт.вик. Клименко О. М. |  | студент 3-го курсу  гр. ІП-51 ФІОТ  Булатов Дмитро Єгорович |

Київ – 2017

**ЗМІСТ:**

[1 Завдання 3](#_Toc495230724)

[2 Виконання Завдання 5](#_Toc495230725)

# Завдання

Завдання 1:

Кожен студент згідно предметної області за варіантом повинен визначити таблицю\_1, що містить оперативні дані, та таблицю\_2, що містить довідник, що використовується в таблиці\_1, скласти наступні види запитів, попередньо описавши словами, що ці запити виконують (запит має бути адекватним, а не формальне застосування наведених службових слів):

1) Використовуючи count() (або будь-яку іншу агрегатну функцію), partition by, order by та запит, що дасть такий самий результат, але не застосовуючи аналітичні функції.

2) Використовуючи rank() або dense\_rank(), partition by, order by та запит, що дасть такий самий результат, але не застосовуючи аналітичні функції.

3) Використовуючи sliding window (rows), partition by, order by та запит, що дасть такий самий результат, але не застосовуючи аналітичні функції.

4) Використовуючи sliding window (range) , partition by, order by та запит, що дасть такий самий результат, але не застосовуючи аналітичні функції.

5) Самостійно розібратися, як застосовується функція lag(). Використовуючи lag(), partition by, order by та запит, що дасть такий самий результат, але не застосовуючи аналітичні функції.

6) Самостійно розібратися, як застосовується функція lead(). Використовуючи lead(), partition by, order by та запит, що дасть такий самий результат, але не застосовуючи аналітичні функції.

Завдання 2 (повторення «Нормалізації» - матеріалу попереднього семестру):

Згідно Вашої предметної області, вибрати таблицю:

1) Змінити таблицю так, щоб вона не була у першій нормальній формі, навести цю таблицю у наповненому вигляді. Потім описати, які зміни треба зробити, щоб вона відповідала першій нормальній формі, навести цю таблицю у наповненому вигляді.

2) Описати, які зміни треба зробити, щоб таблиця відповідала другій нормальній формі, навести цю таблицю у наповненому вигляді.

3) Описати, які зміни треба зробити, щоб таблиця відповідала третій нормальній формі, навести цю таблицю у наповненому вигляді.

4) Описати, які зміни треба зробити, щоб таблиця відповідала нормальній формі Бойса-Кода, навести цю таблицю у наповненому вигляді.

**Преметна область**: Навчання з охорони праці

# Виконання Завдання

1. Завдання по написанню запитів
   1. Використовуючи count() (або будь-яку іншу агрегатну функцію), partition by, order by та запит, що дасть такий самий результат, але не застосовуючи аналітичні функції:  
      *Призначення*: Підраховує кількість різних адрес користувачів

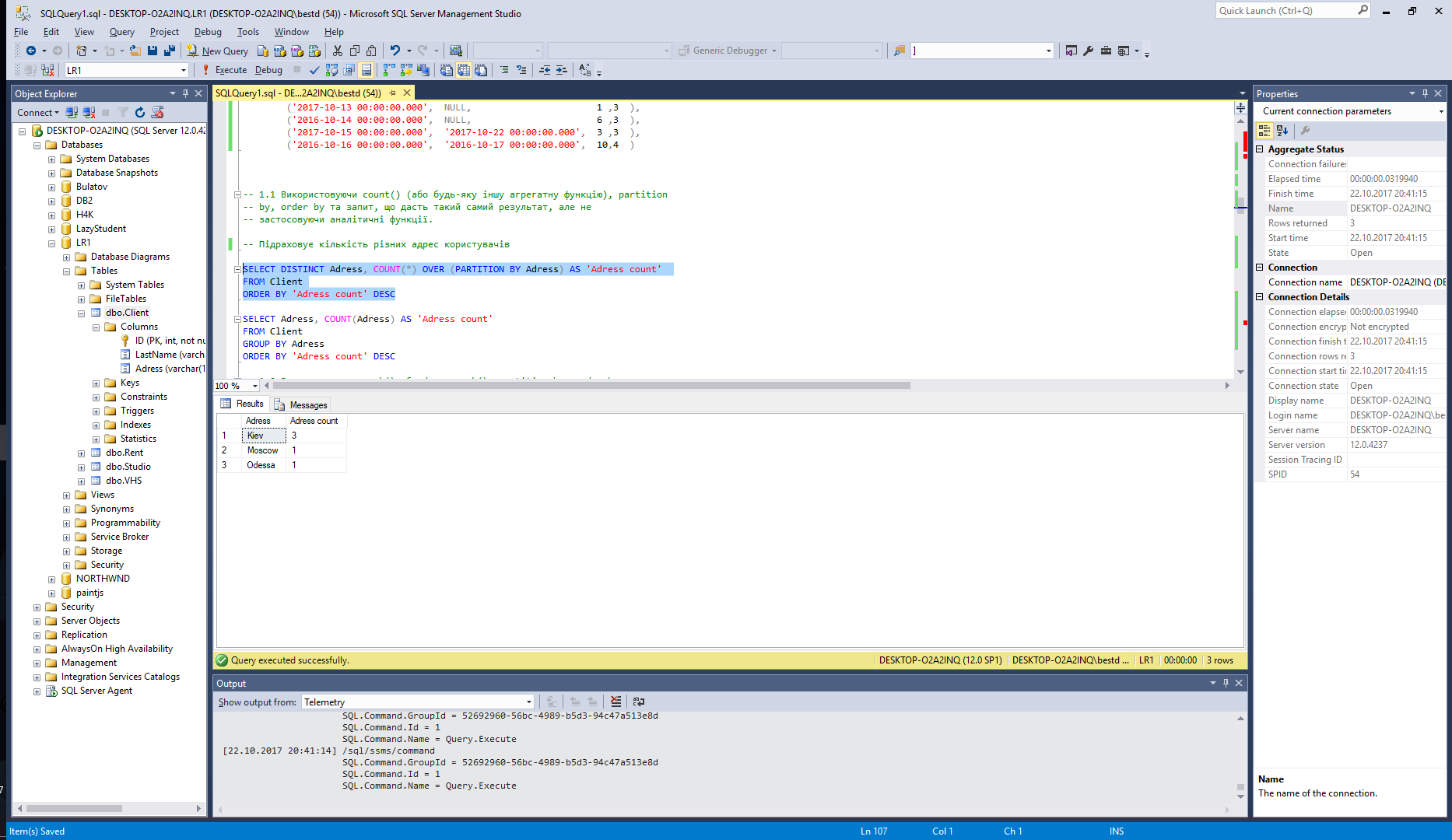
*Текст запиту з використанням аналітичних функцій*:

SELECT DISTINCT Adress, COUNT(\*) OVER (PARTITION BY Adress) AS 'Adress count'

FROM Client

ORDER BY 'Adress count' DESC

*Результат виконання з використання аналітичних функцій*:



*Текст запиту без використанням аналітичних функцій*:

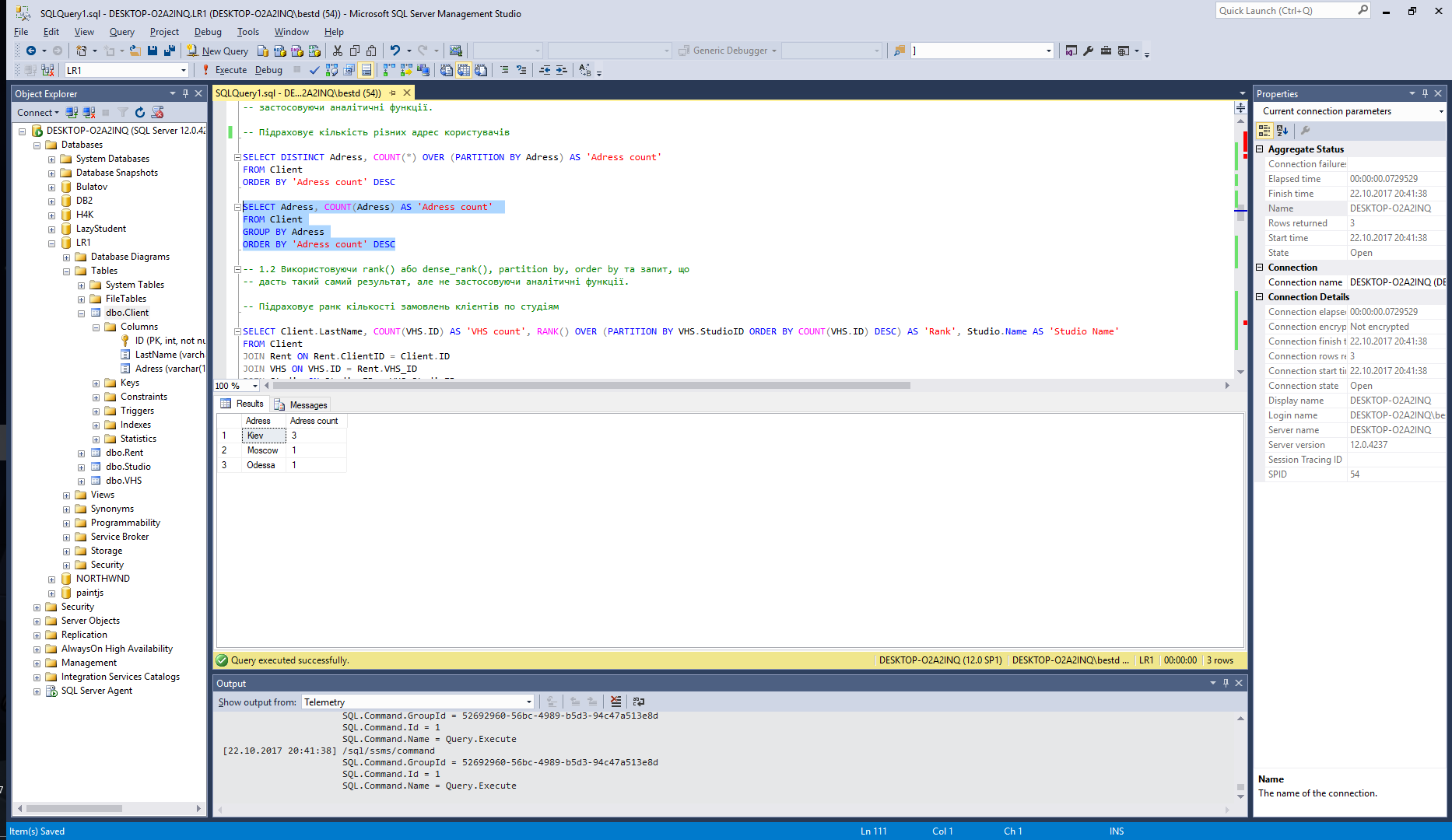
SELECT Adress, COUNT(Adress) AS 'Adress count'

FROM Client

GROUP BY Adress

ORDER BY 'Adress count' DESC

*Результат виконання без використання аналітичних функцій*:



* 1. Використовуючи rank() або dense\_rank(), partition by, order by та запит, що дасть такий самий результат, але не застосовуючи аналітичні функції:

*Призначення*: Підраховує ранк кількості замовлень клієнтів по студіям

*Текст запиту з використанням аналітичних функцій*:

SELECT Client.LastName, COUNT(VHS.ID) AS 'VHS count', RANK() OVER (PARTITION BY VHS.StudioID ORDER BY COUNT(VHS.ID) DESC) AS 'Rank', Studio.Name AS 'Studio Name'

FROM Client

JOIN Rent ON Rent.ClientID = Client.ID

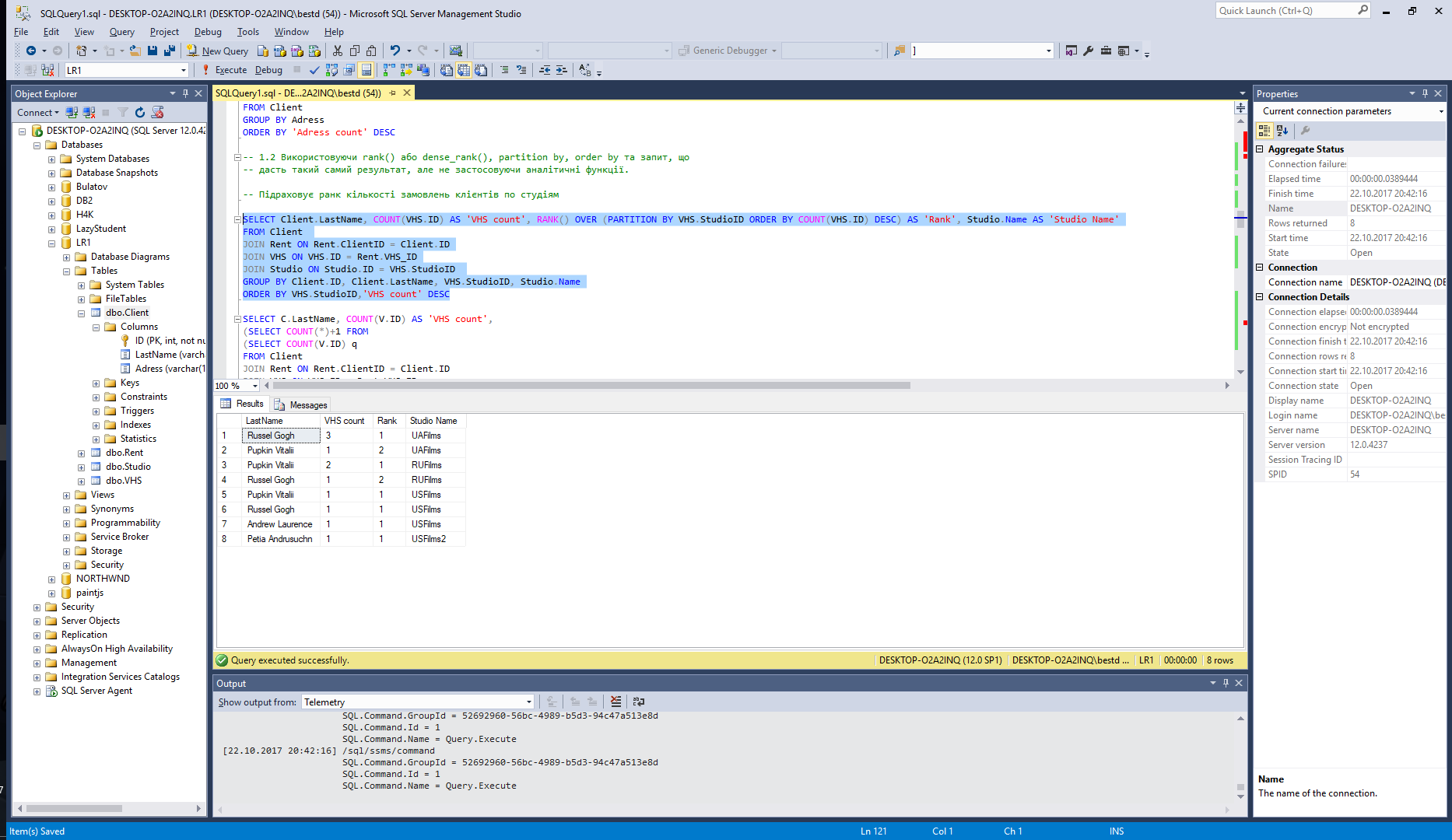
JOIN VHS ON VHS.ID = Rent.VHS\_ID

JOIN Studio ON Studio.ID = VHS.StudioID

GROUP BY Client.ID, Client.LastName, VHS.StudioID, Studio.Name

ORDER BY VHS.StudioID,'VHS count' DESC

*Результат виконання з використання аналітичних функцій*:



*Текст запиту без використанням аналітичних функцій*:

SELECT C.LastName, COUNT(V.ID) AS 'VHS count',

(SELECT COUNT(\*)+1 FROM

(SELECT COUNT(V.ID) q

FROM Client

JOIN Rent ON Rent.ClientID = Client.ID

JOIN VHS ON VHS.ID = Rent.VHS\_ID

WHERE VHS.StudioID = V.StudioID GROUP BY ClientID HAVING COUNT(V.ID)<COUNT(VHS.ID) )

e) AS 'Rank',

S.Name AS 'Studio Name'

FROM Client AS C

JOIN Rent AS R ON R.ClientID = C.ID

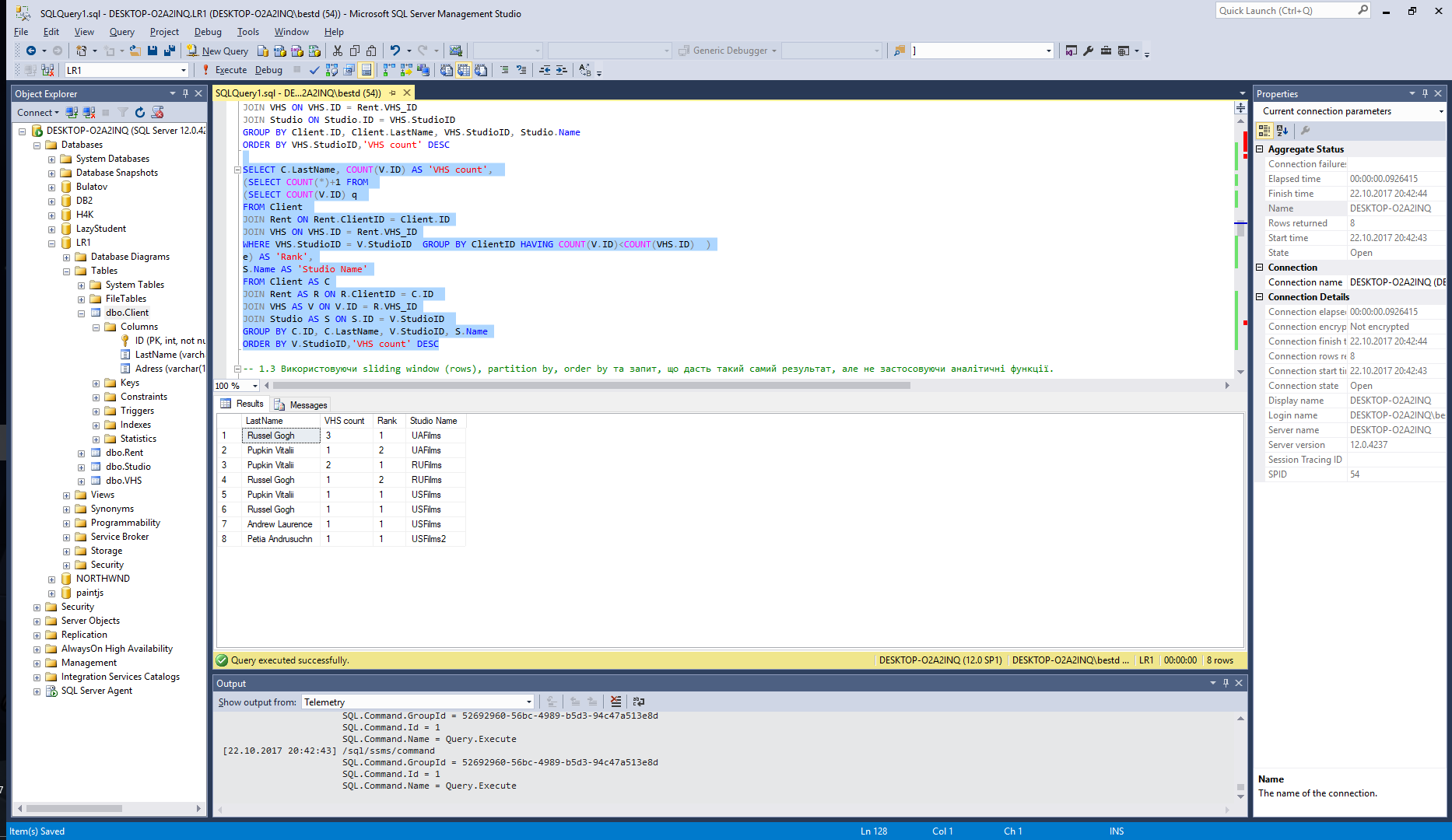
JOIN VHS AS V ON V.ID = R.VHS\_ID

JOIN Studio AS S ON S.ID = V.StudioID

GROUP BY C.ID, C.LastName, V.StudioID, S.Name

ORDER BY V.StudioID,'VHS count' DESC

*Результат виконання без використання аналітичних функцій*:



* 1. Використовуючи sliding window (rows), partition by, order by та запит, що дасть такий самий результат, але не застосовуючи аналітичні функції

*Призначення*: Виводить кількість повернутих касет у суміжних замовленнях клієнта

*Текст запиту з використанням аналітичних функцій*:

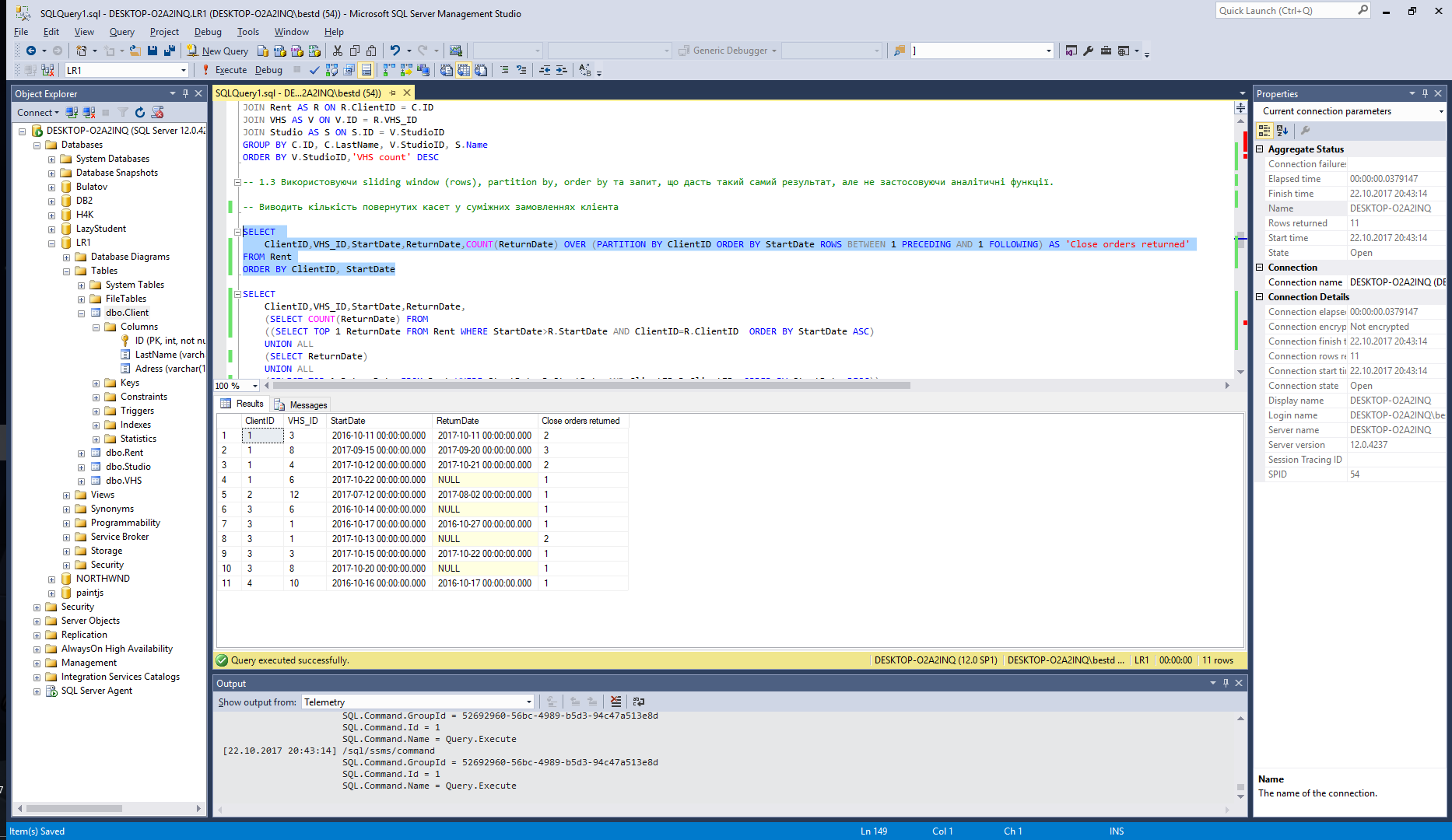
SELECT

ClientID,VHS\_ID,StartDate,ReturnDate,COUNT(ReturnDate) OVER (PARTITION BY ClientID ORDER BY StartDate ROWS BETWEEN 1 PRECEDING AND 1 FOLLOWING) AS 'Close orders returned'

FROM Rent

ORDER BY ClientID, StartDate

*Результат виконання з використання аналітичних функцій*:



*Текст запиту без використанням аналітичних функцій*:

SELECT

ClientID,VHS\_ID,StartDate,ReturnDate,

(SELECT COUNT(ReturnDate) FROM

((SELECT TOP 1 ReturnDate FROM Rent WHERE StartDate>R.StartDate AND ClientID=R.ClientID ORDER BY StartDate ASC)

UNION ALL

(SELECT ReturnDate)

UNION ALL

(SELECT TOP 1 ReturnDate FROM Rent WHERE StartDate<R.StartDate AND ClientID=R.ClientID ORDER BY StartDate DESC))

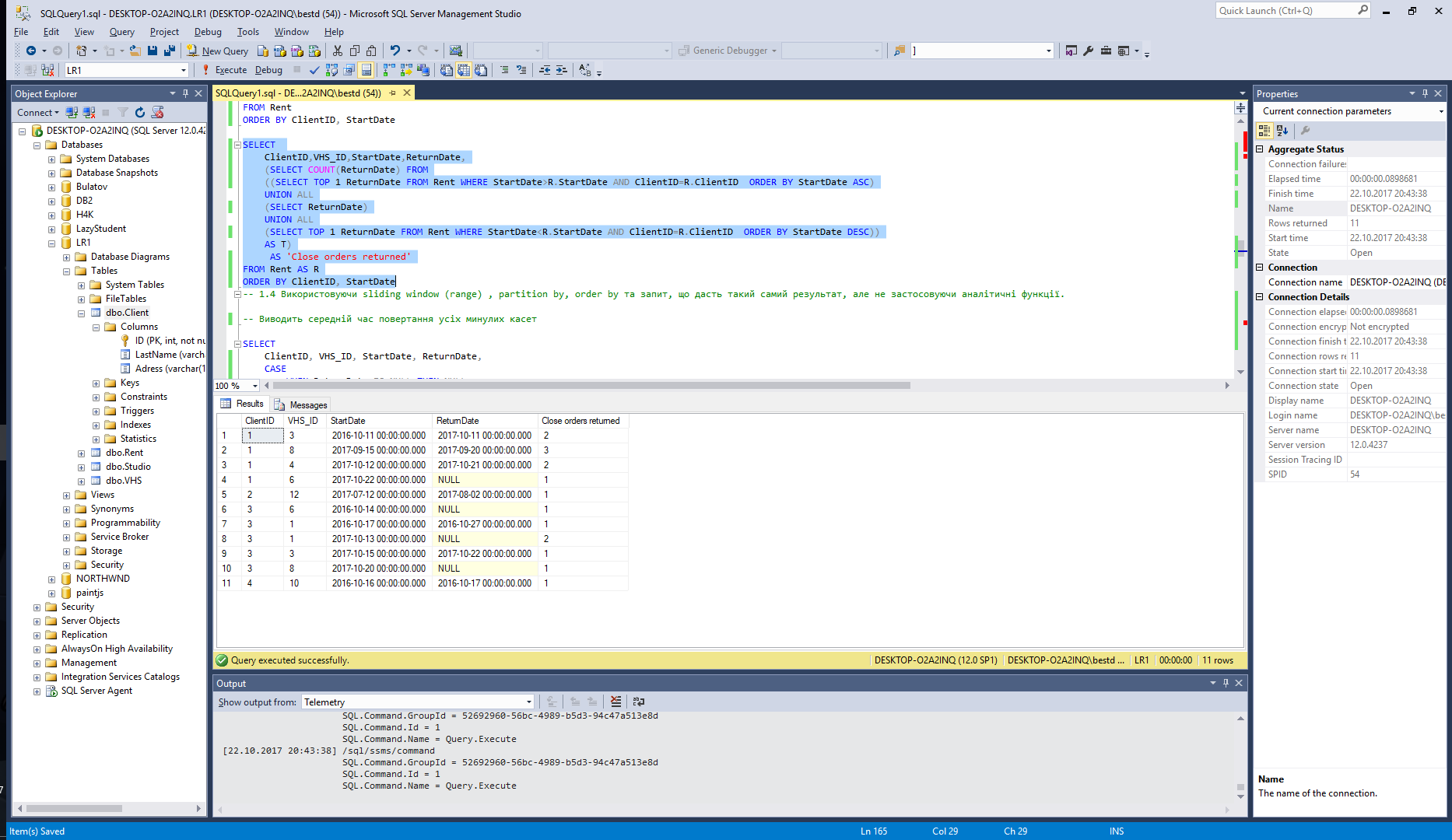
AS T)

AS 'Close orders returned'

FROM Rent AS R

ORDER BY ClientID, StartDate

*Результат виконання без використання аналітичних функцій*:



* 1. Використовуючи sliding window (range) , partition by, order by та запит, що дасть такий самий результат, але не застосовуючи аналітичні функції.:

*Призначення*: Виводить середній час повертання усіх минулих касет конкретним користувачем

*Текст запиту з використанням аналітичних функцій*:

SELECT

ClientID, VHS\_ID, StartDate, ReturnDate,

CASE

WHEN ReturnDate IS NULL THEN NULL

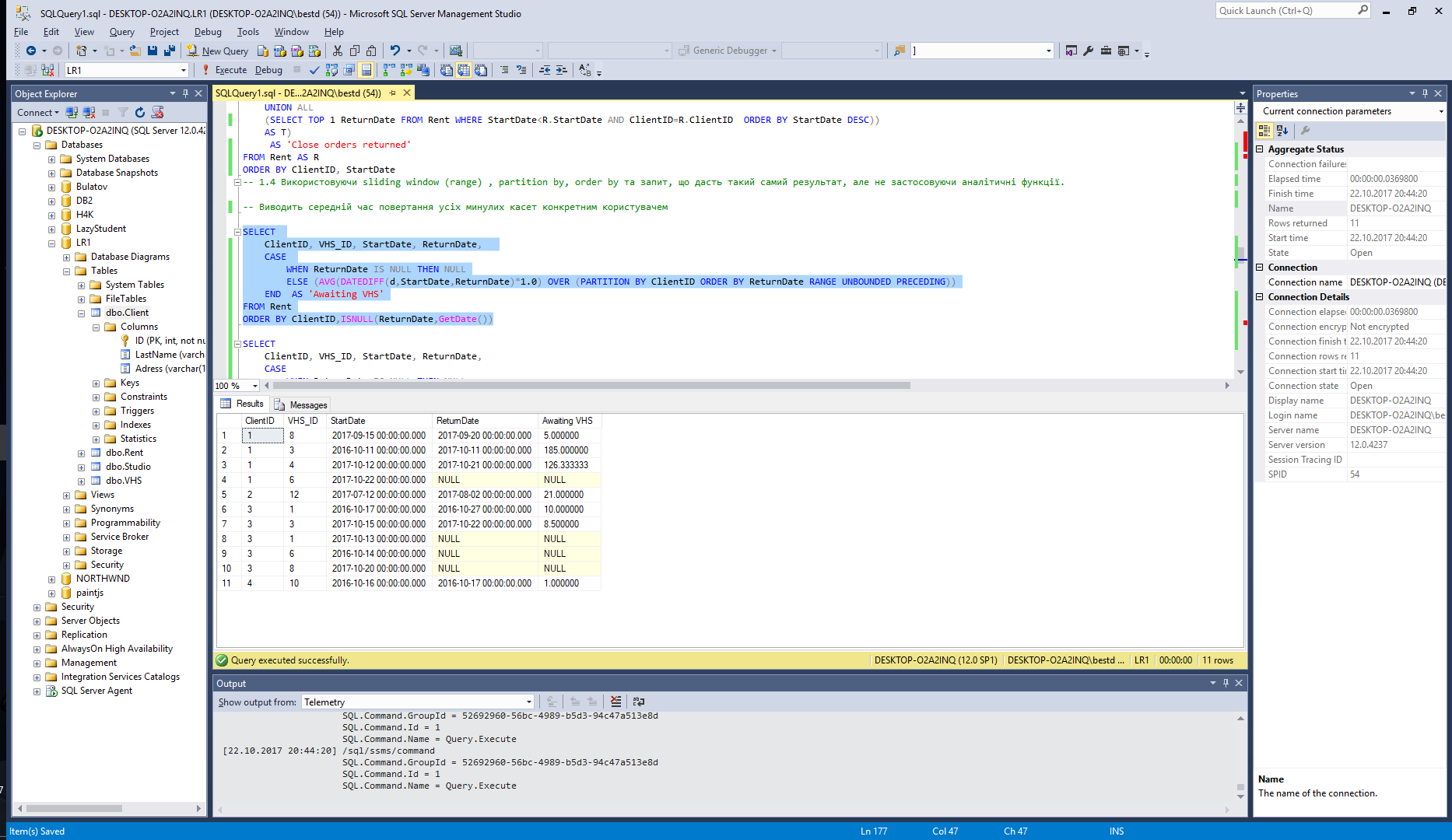
ELSE (AVG(DATEDIFF(d,StartDate,ReturnDate)\*1.0) OVER (PARTITION BY ClientID ORDER BY ReturnDate RANGE UNBOUNDED PRECEDING))

END AS 'Awaiting VHS'

FROM Rent

ORDER BY ClientID,ISNULL(ReturnDate,GetDate())

*Результат виконання з використання аналітичних функцій*:



*Текст запиту без використанням аналітичних функцій*:

SELECT

ClientID, VHS\_ID, StartDate, ReturnDate,

CASE

WHEN ReturnDate IS NULL THEN NULL

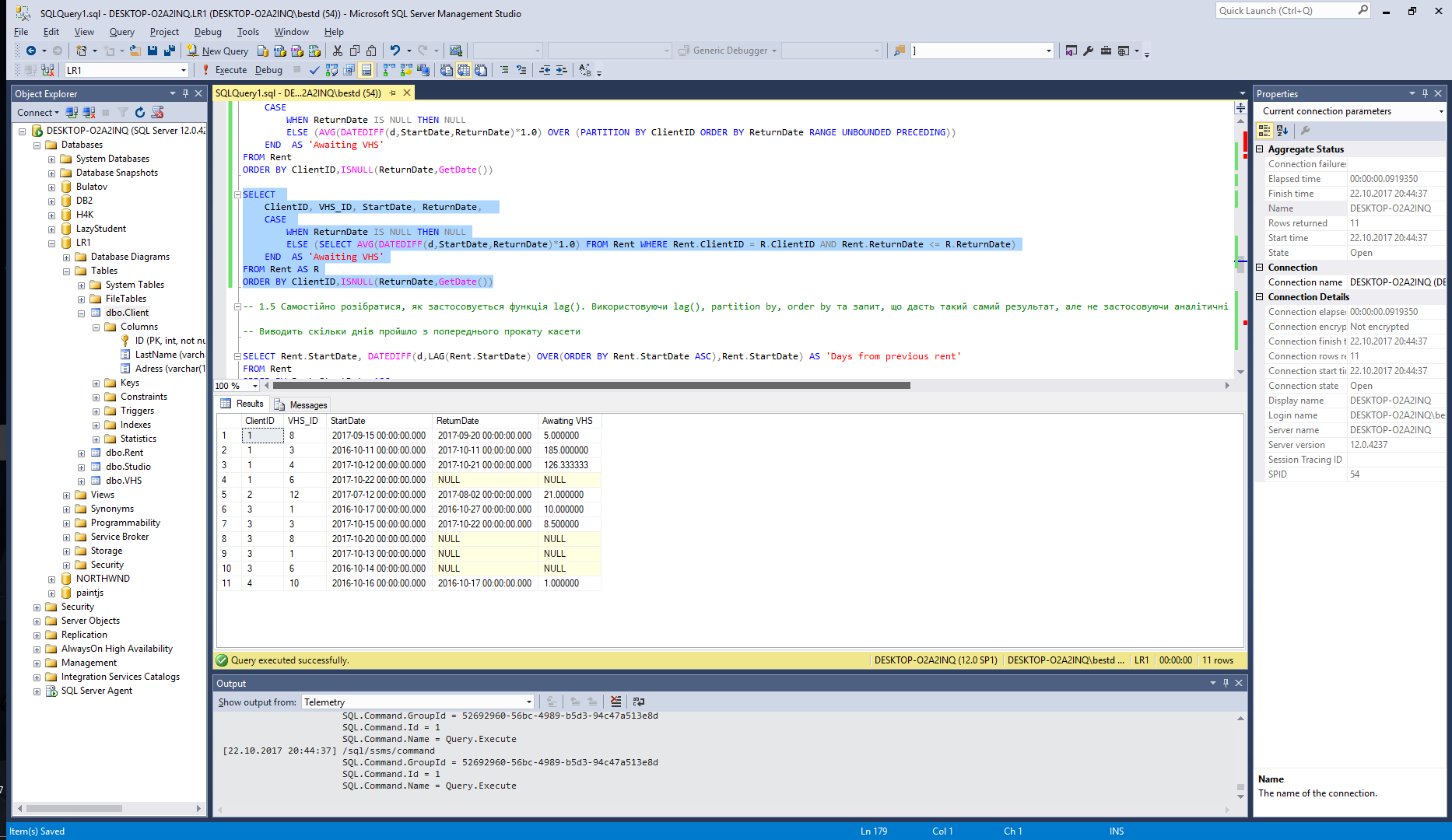
ELSE (SELECT AVG(DATEDIFF(d,StartDate,ReturnDate)\*1.0) FROM Rent WHERE Rent.ClientID = R.ClientID AND Rent.ReturnDate <= R.ReturnDate)

END AS 'Awaiting VHS'

FROM Rent AS R

ORDER BY ClientID,ISNULL(ReturnDate,GetDate())

*Результат виконання без використання аналітичних функцій*:



* 1. Самостійно розібратися, як застосовується функція lag(). Використовуючи lag(), partition by, order by та запит, що дасть такий самий результат, але не застосовуючи аналітичні функції:

*Призначення*: Виводить скільки днів пройшло з попереднього прокату касети клієнтом.

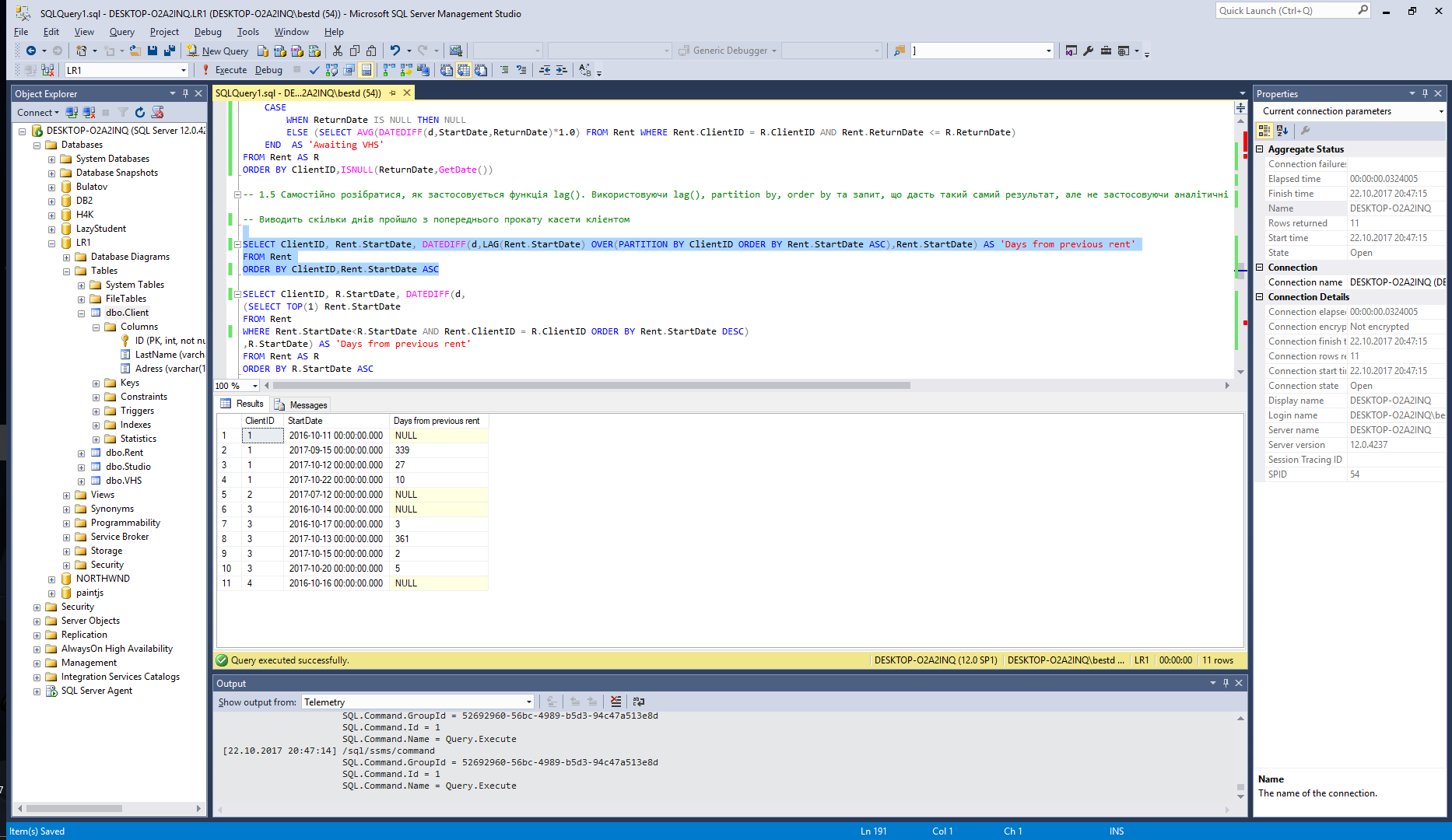
*Текст запиту з використанням аналітичних функцій*:

SELECT ClientID, Rent.StartDate, DATEDIFF(d,LAG(Rent.StartDate) OVER(PARTITION BY ClientID ORDER BY Rent.StartDate ASC),Rent.StartDate) AS 'Days from previous rent'

FROM Rent

ORDER BY ClientID,Rent.StartDate ASC

*Результат виконання з використання аналітичних функцій*:



*Текст запиту без використанням аналітичних функцій*:

SELECT ClientID, R.StartDate, DATEDIFF(d,

(SELECT TOP(1) Rent.StartDate

FROM Rent

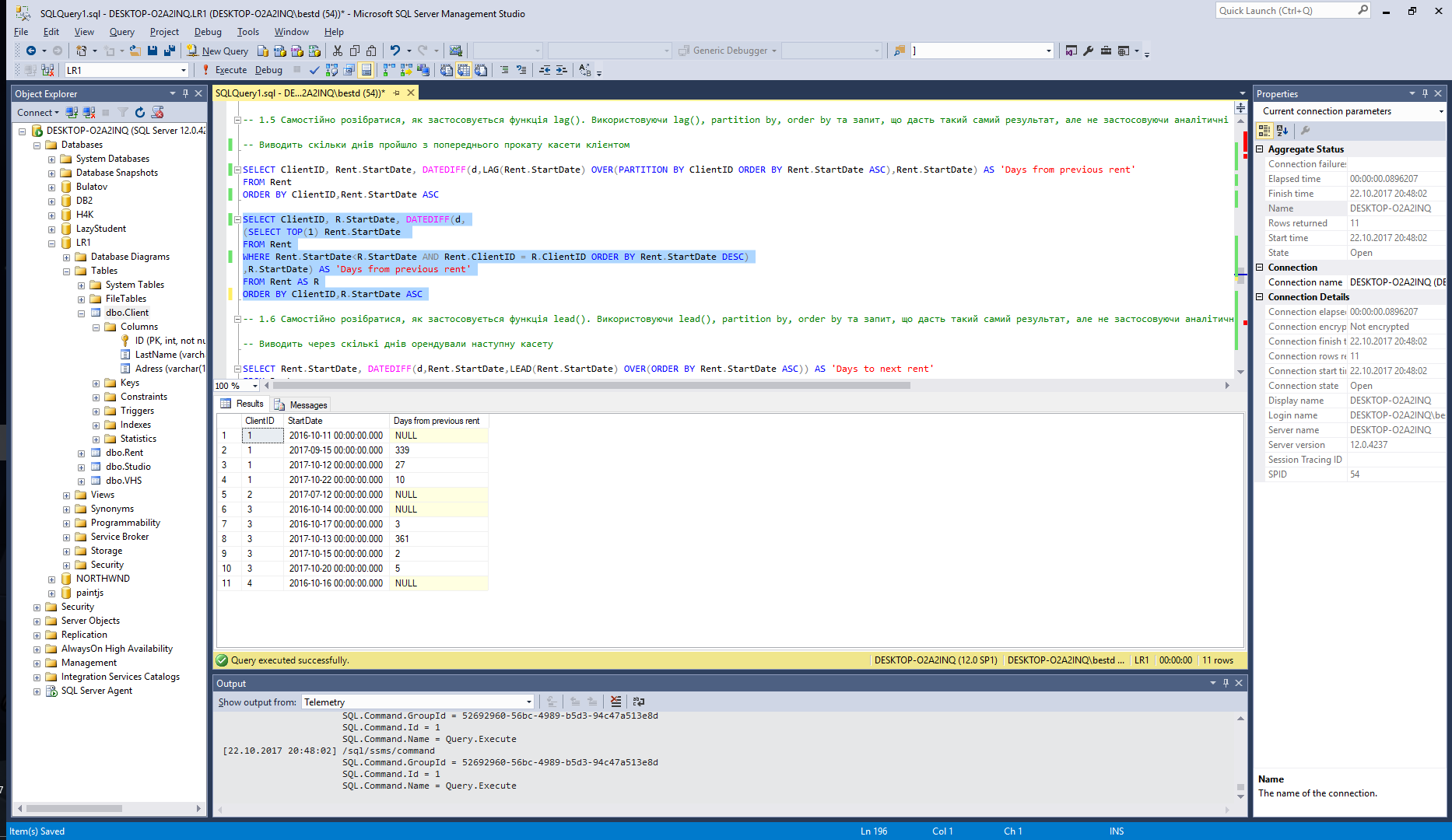
WHERE Rent.StartDate<R.StartDate AND Rent.ClientID = R.ClientID ORDER BY Rent.StartDate DESC)

,R.StartDate) AS 'Days from previous rent'

FROM Rent AS R

ORDER BY ClientID,R.StartDate ASC

*Результат виконання без використання аналітичних функцій*:



* 1. Самостійно розібратися, як застосовується функція lead(). Використовуючи lead(), partition by, order by та запит, що дасть такий самий результат, але не застосовуючи аналітичні функції:

*Призначення*: Для кожного слухача та кожного придбаного курсу вивести вартість поточного та хронологічно наступного пройденого курса.

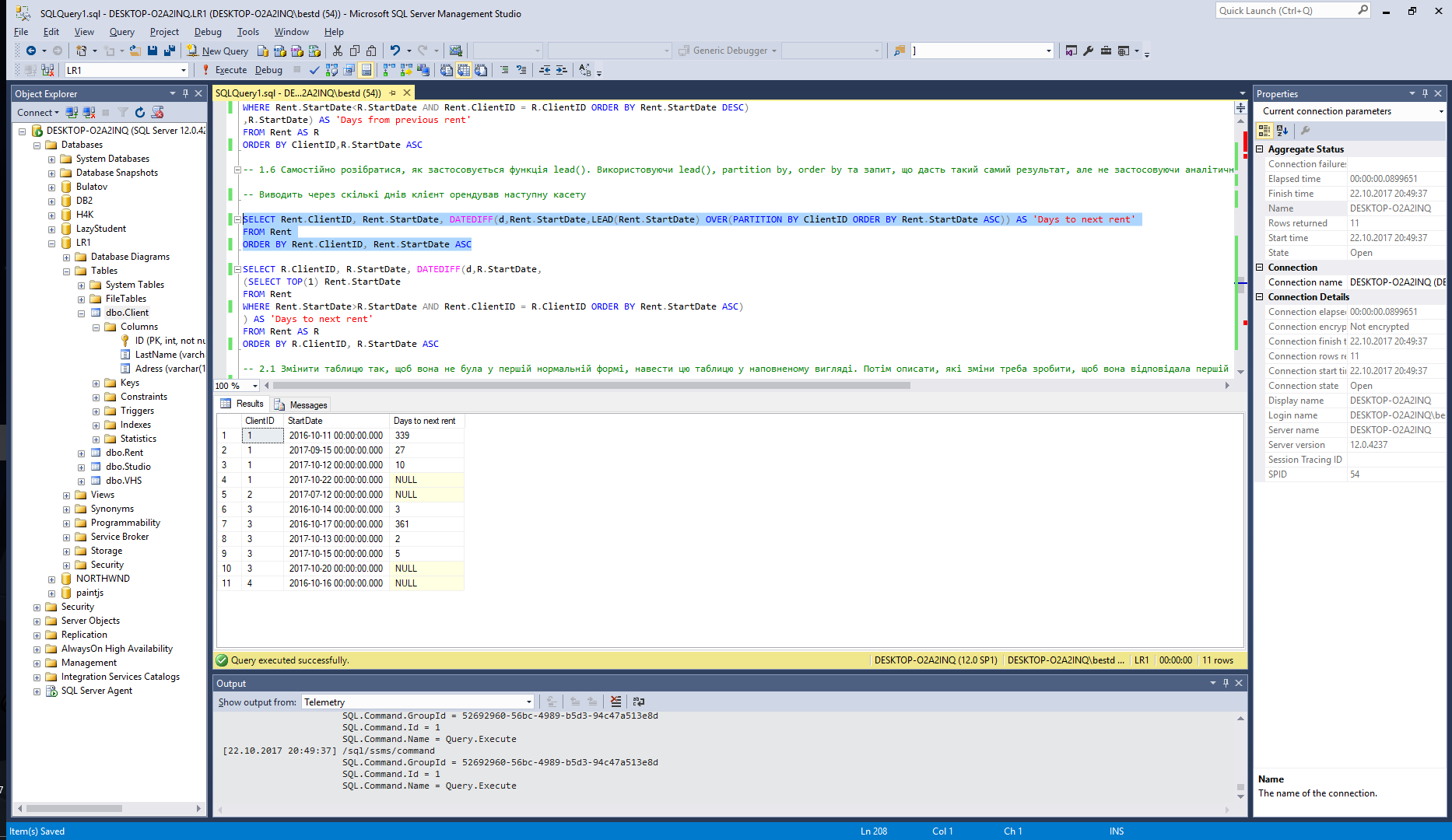
*Текст запиту з використанням аналітичних функцій*:

SELECT Rent.ClientID, Rent.StartDate, DATEDIFF(d,Rent.StartDate,LEAD(Rent.StartDate) OVER(PARTITION BY ClientID ORDER BY Rent.StartDate ASC)) AS 'Days to next rent'

FROM Rent

ORDER BY Rent.ClientID, Rent.StartDate ASC

*Результат виконання з використання аналітичних функцій*:



*Текст запиту без використанням аналітичних функцій*:

SELECT R.ClientID, R.StartDate, DATEDIFF(d,R.StartDate,

(SELECT TOP(1) Rent.StartDate

FROM Rent

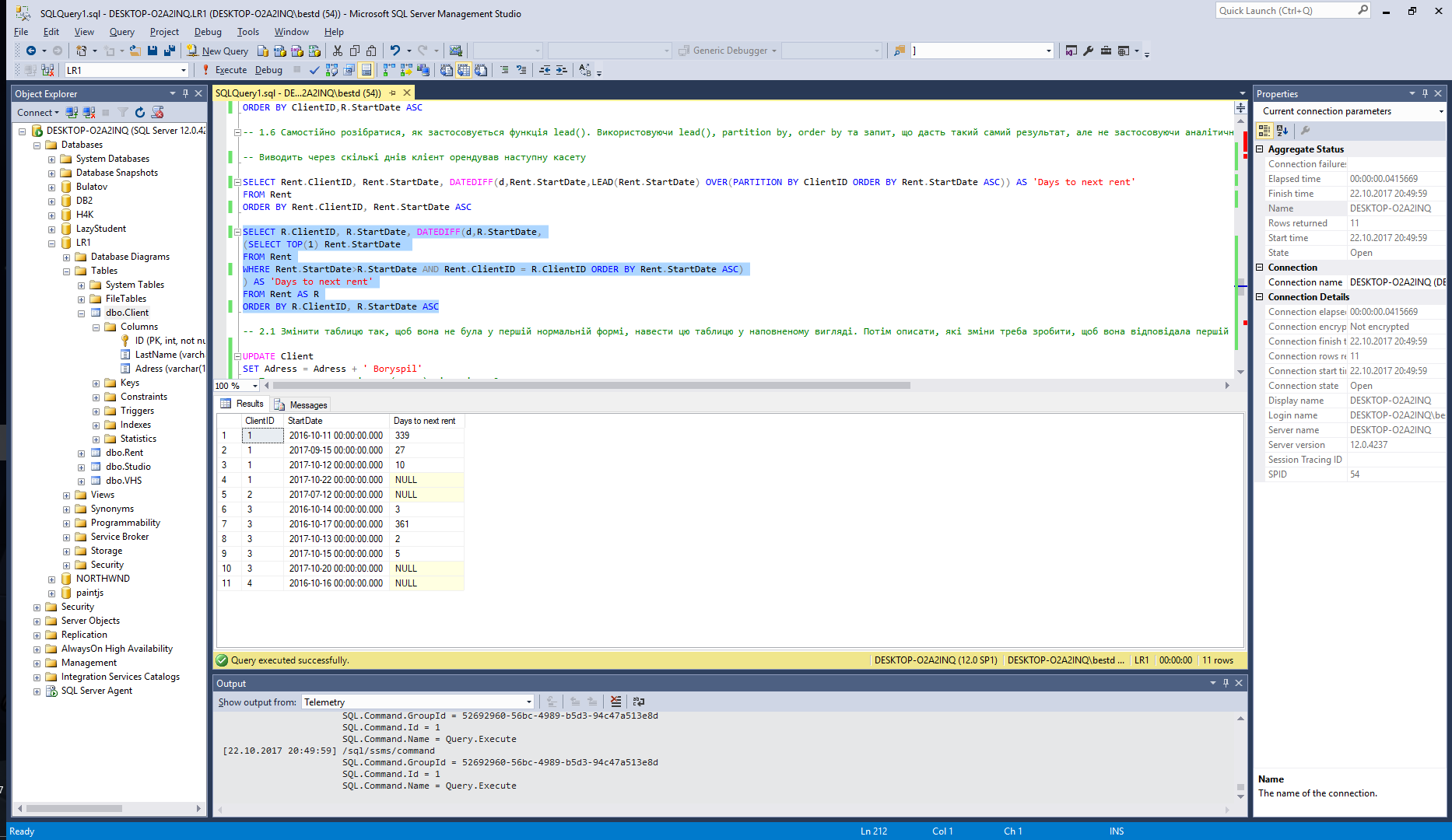
WHERE Rent.StartDate>R.StartDate AND Rent.ClientID = R.ClientID ORDER BY Rent.StartDate ASC)

) AS 'Days to next rent'

FROM Rent AS R

ORDER BY R.ClientID, R.StartDate ASC

*Результат виконання без використання аналітичних функцій*:



1. Завдання по нормалізації БД:
   1. Змінити таблицю так, щоб вона не була у першій нормальній формі, навести цю таблицю у наповненому вигляді. Потім описати, які зміни треба зробити, щоб вона відповідала першій нормальній формі, навести цю таблицю у наповненому вигляді.

CREATE TABLE Client(

ID int NOT NULL PRIMARY KEY IDENTITY(1,1),

LastName varchar(40),

Adress varchar(100),

Birthday varchar(100),

RegDate datetime DEFAULT(GetDate())

)

INSERT INTO [dbo].[Client]

(LastName,Adress,Birthday)

VALUES

('Pupkin Vitalii' ,'Kiev' ,'30 June' ),

('Petia Andrusuchn' ,'Moscow','15 July' ),

('Russel Gogh' ,'Kiev' ,'10 August' ),

('Andrew Laurence' ,'Odessa','23 September'),

('Petia Lait' ,'Kiev' ,'12 December' )

UPDATE Client

SET Adress = Adress + ' Boryspil'

-- Тепер у кожному клієнту (рядку) відповідає 2 адреси

SELECT \* FROM Client Order BY ID;

-- Треба щоб залишилась тільки 1 відповідність на рядок

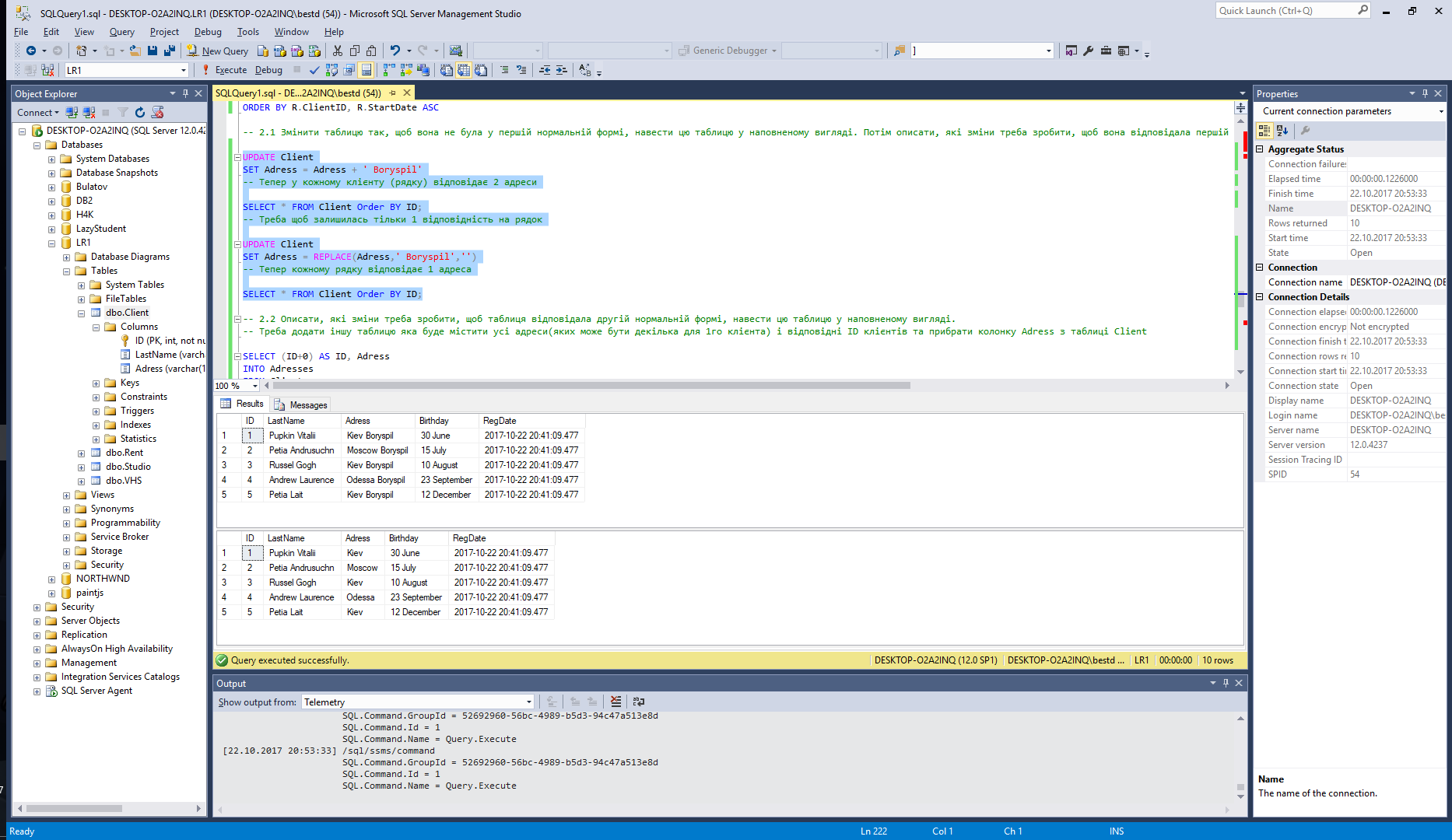
UPDATE Client

SET Adress = REPLACE(Adress,' Boryspil','')

-- Тепер кожному рядку відповідає 1 адреса

SELECT \* FROM Client Order BY ID;

*Результат виконання:*



* 1. Описати, які зміни треба зробити, щоб таблиця відповідала другій нормальній формі, навести цю таблицю у наповненому вигляді*.*

*Текст запиту*:

-- Треба додати іншу таблицю яка буде містити усі адреси(яких може бути декілька для 1го клієнта) і відповідні ID клієнтів та прибрати колонку Adress з таблиці Client

SELECT (ID+0) AS ID, Adress

INTO Adresses

FROM Client

INSERT INTO Adresses

SELECT ID, 'Boryspil'

FROM Client

ALTER TABLE Client

DROP COLUMN Adress

-- Тепер данні про адреси що можуть повторюватись зберігаються у окремій таблиці

SELECT \*

FROM Client

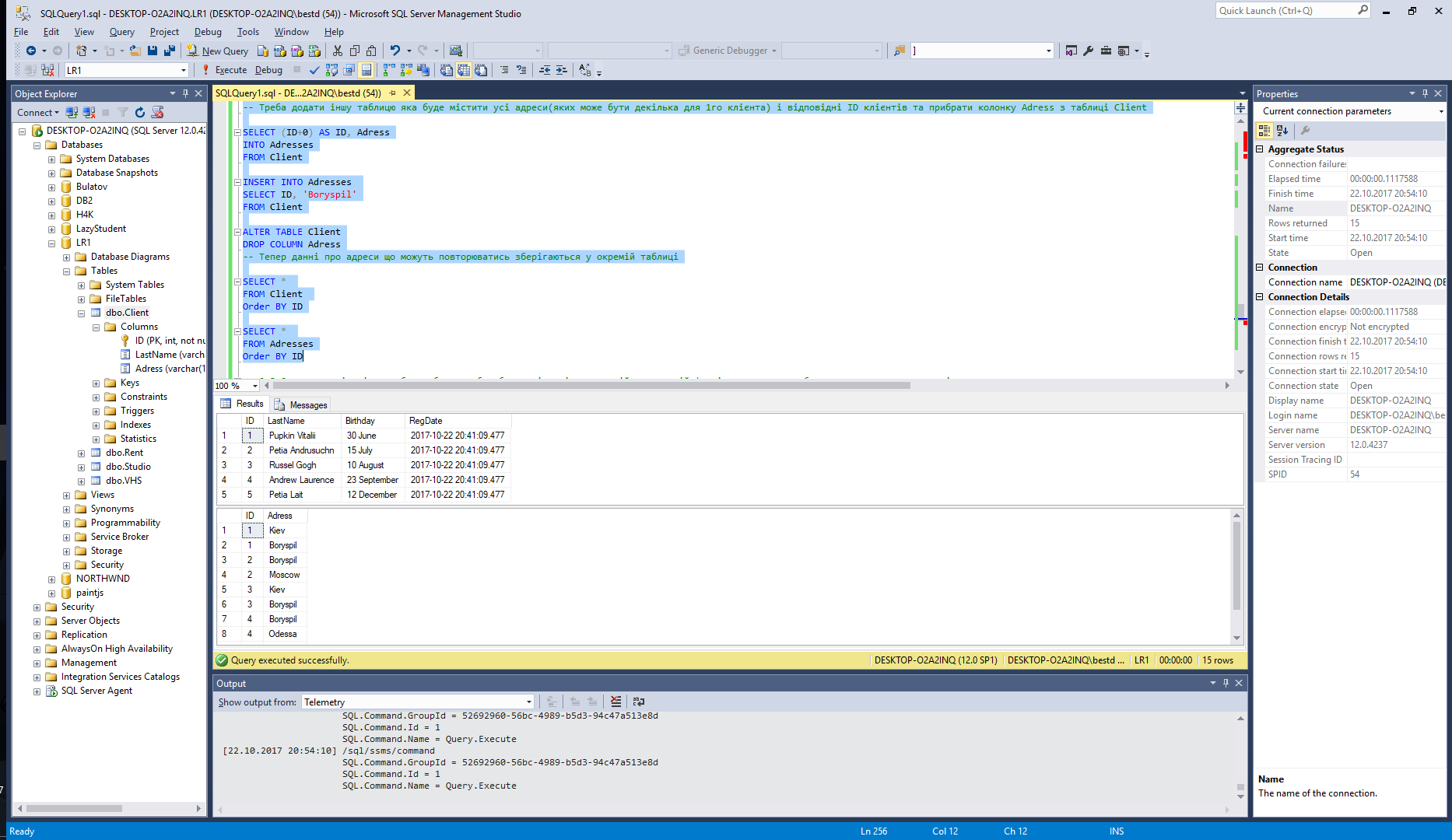
Order BY ID

SELECT \*

FROM Adresses

Order BY ID

*Результат виконання:*



* 1. Описати, які зміни треба зробити, щоб таблиця відповідала третій нормальній формі, навести цю таблицю у наповненому вигляді.

*Текст запиту*:

-- Треба винести усі колонки що залежать не тільки від первинного ключа у окремі таблиці

SELECT LastName, Birthday

INTO ClientBirthdays

FROM Client

ALTER TABLE Client

DROP COLUMN Birthday

-- Тепер нема колонок що залежать не тільки від первинного ключа

SELECT \*

FROM Client

Order BY ID

SELECT \*

FROM Adresses

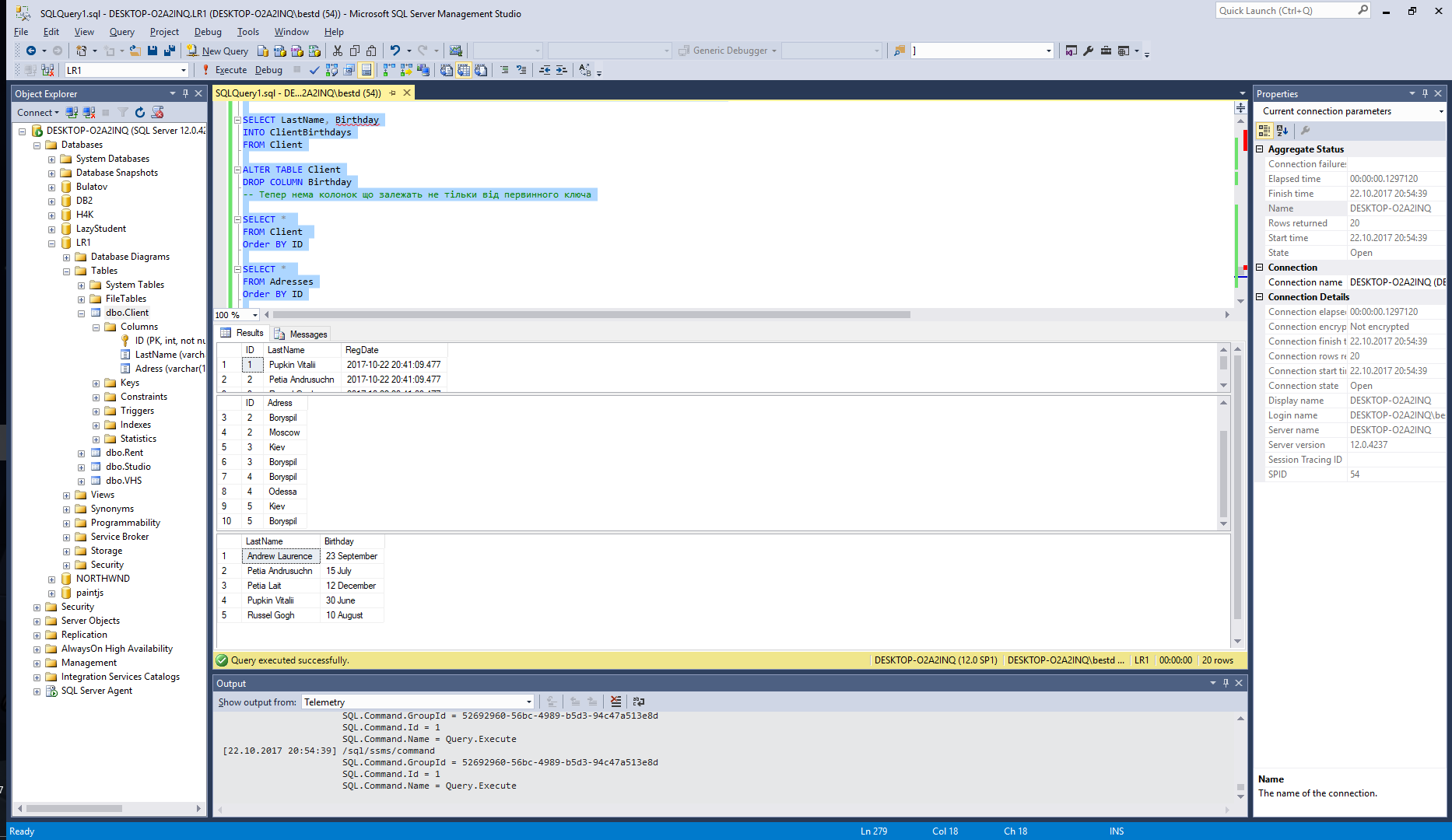
Order BY ID

SELECT \*

FROM ClientBirthdays

Order BY LastName

*Результат виконання:*



* 1. Описати, які зміни треба зробити, щоб таблиця відповідала нормальній формі Бойса-Кода, навести цю таблицю у наповненому вигляді*.*

*Текст запиту*:

-- Таблиця уже відповідає нормальній формі Бойса-Кода так як в ній не залишилось функціональних залежностей не винесених в окрему таблицю

SELECT \*

FROM Client

Order BY ID

SELECT \*

FROM Adresses

Order BY ID

SELECT \*

FROM ClientBirthdays

Order BY LastName

*Результат виконання:*

